IN THE CLAIMS

No amendments are being made in this response. The pending claims are as follows:

1. A computer-implemented method, comprising:

analyzing source code to determine a program slice;

creating a program slice diagram that provides a graphical representation of the program slice; and

displaying the program slice diagram.

2. The method of claim 1, wherein the program slice diagram further comprises a directed graph comprising a plurality of nodes and arcs.

3. The method of claim 2, wherein the arcs represent data flow dependencies between the nodes.

4. The method of claim 2, wherein the nodes represent source code statements within a selected subroutine.

5. The method of claim 2, wherein the nodes represent variable references outside of a selected subroutine.

6. The method of claim 2, wherein the nodes represent calls made to subroutines.

7. The method of claim 3, further comprising:

labeling the arcs with variable names.

8. The method of claim 1, further comprising

pruning the program slice diagram.

9. The method of claim 1, wherein the program slice diagram is displayed in multi-column layout.

10. The method of claim 1, wherein the program slice diagram is displayed in upside-down-tree layout.

11. The method of claim 1, further comprising:

in response to the analyzing element, performing semantic abstraction to group a subset of the nodes together based on semantic information.

12. The method of claim 11, further comprising:

identifying a logical category of computations; and

displaying the logical category of computations with a cross-reference to a display of the source code.

13. The method of claim 11, wherein performing semantic abstraction further comprises:

identifying a logical category of data elements; and

displaying the logical category of data elements with a cross-reference to a display of data.

14. The method of claim 2, further comprising:

moving the nodes and arcs in a response to a user request to rearrange the program slice diagram.

15. The method of claim 1, further comprising:

panning the program slice diagram.

16. The method of claim 1, further comprising:

zooming the program slice diagram.

17. The method of claim 1, further comprising:

pruning a subgraph of the program slice diagram.

18. The method of claim 2, further comprising:

collapsing a subgraph of the program slice diagram into a node.

19. The method of claim 2, further comprising:

simplifying the program slice diagram by retaining only those nodes that correspond to variable references outside of a selected subroutine.

20. The method of claim 1, further comprising:

performing event abstraction by collapsing together nodes that correspond to a logical event.

21. The method of claim 1, further comprising:

displaying the program slice diagram with a cross-reference to a portion of the source code that is associated with the program slice diagram.

22. A signal-bearing media comprising computer-readable instructions, wherein the instructions when read and executed by a computer comprise:

analyzing source code to determine a program slice;

creating a program slice diagram that provides a graphical representation of the program slice; and

displaying the program slice diagram.

23. The signal-bearing media of claim 22, wherein the program slice diagram further comprises a directed graph comprising a plurality of nodes and arcs.

24. The signal-bearing media of claim 23, wherein the arcs represent data flow dependencies between the nodes.

25. The signal-bearing media of claim 23, wherein the nodes represent source code statements within a selected subroutine.

26. The signal-bearing media of claim 23, wherein the nodes represent variable references outside of a selected subroutine.

27. The signal-bearing media of claim 23, wherein the nodes represent calls made to subroutines.

28. The signal-bearing media of claim 23, further comprising:

labeling the arcs with variable names.

29. The signal-bearing media of claim 22, further comprising

pruning the program slice diagram.

30. The signal-bearing media of claim 22, wherein the program slice diagram is displayed in multi-column layout.

31. The signal-bearing media of claim 22, wherein the program slice diagram is displayed in upside-down-tree layout.

32. The signal-bearing media of claim 22, further comprising:

in response to the analyzing element, performing semantic abstraction to group a subset of the nodes together based on semantic information.

33. The signal-bearing media of claim 32, further comprising:

identifying a logical category of computations; and

displaying the logical category of computations with a cross-reference to a display of the source code.

34. The signal-bearing media of claim 32, wherein performing semantic abstraction further comprises:

identifying a logical category of data elements; and

displaying the logical category of data elements with a cross-reference to a display of data.

35. The signal-bearing media of claim 23, further comprising:

moving the nodes and arcs in a response to a user request to rearrange the program slice diagram.

36. The signal-bearing media of claim 22, further comprising:

panning the program slice diagram.

37. The signal-bearing media of claim 22, further comprising:

zooming the program slice diagram.

38. The signal-bearing media of claim 22, further comprising:

pruning a subgraph of the program slice diagram.

39. The signal-bearing media of claim 22, further comprising:

collapsing a subgraph of the program slice diagram into a node.

40. The signal-bearing media of claim 22, further comprising:

simplifying the program slice diagram by retaining only those nodes that correspond to variable references that are outside of a selected subroutine.

41. The signal-bearing media of claim 22, further comprising:

performing event abstraction by collapsing together nodes that correspond to a logical event.

42. The signal-bearing media of claim 22, further comprising:

displaying the program slice diagram with a cross-reference to a portion of the source code that is associated with the program slice diagram.

43. A computer system, comprising:

a processor; and

a memory coupled to the processor wherein the memory comprises instructions executable by the processor, wherein the instructions when executed by the processor comprise:

analyzing source code to determine a program slice,

creating a program slice diagram that provides a graphical representation of the program slice, and

displaying the program slice diagram.

44. The computer system of claim 43, wherein the program slice diagram further comprises a directed graph comprising a plurality of nodes and arcs.

45. The computer system of claim 44, wherein the arcs represent data flow dependencies between the nodes.

46. The computer system of claim 44, wherein the nodes represent source code statements within a selected subroutine. .

47. The computer system of claim 44, wherein the nodes represent variable references outside of a selected subroutine.

48. The computer system of claim 44, wherein the nodes represent calls made to subroutines.

49. The computer system of claim 43, wherein the instructions further comprise:

labeling the arcs with variable names.

50. The computer system of claim 43, wherein the instructions further comprise

pruning the program slice diagram.

51. The computer system of claim 43, wherein the program slice diagram is displayed in multi-column layout.

52. The computer system of claim 43, wherein the program slice diagram is displayed in upside-down-tree layout.

53. The computer system of claim 43, wherein the instructions further comprise:

in response to the analyzing element, performing semantic abstraction to group a subset of the nodes together based on semantic information.

54. The computer system of claim 53, wherein the instructions further comprise:

identifying a logical category of computations; and

displaying the logical category of computations with a cross-reference to a display of the source code.

55. The computer system of claim 53, wherein performing semantic abstraction further comprises:

identifying a logical category of data elements; and

displaying the logical category of data elements with a cross-reference to a display of data.

56. The computer system of claim 44, wherein the instructions further comprise:

moving the nodes and arcs in a response to a user request to rearrange the program slice diagram.

57. The computer system of claim 43, wherein the instructions further comprise:

panning the program slice diagram.

58. The computer system of claim 43, wherein the instructions further comprise:

zooming the program slice diagram.

59. The computer system of claim 43, wherein the instructions further comprise:

pruning a subgraph of the program slice diagram.

60. The computer system of claim 43, wherein the instructions further comprise:

collapsing a subgraph of the program slice diagram into a node.

61. The computer system of claim 43, wherein the instructions further comprise:

simplifying the program slice diagram by retaining only those nodes that correspond to variable references outside of a selected subroutine.

62. The computer system of claim 43, wherein the instructions further comprise:

performing event abstraction by collapsing together nodes that correspond to a logical event.

63. The computer system of claim 43, wherein the instructions further comprise:

displaying the program slice diagram with a cross-reference to a portion of the source code that is associated with the program slice diagram.

64. A software visualization environment for visualizing source code, comprising:

a code browser;

a block-level abstract syntax tree viewer;

a program slice browser; and

a template viewer.

65. The software visualization environment of claim 64, wherein the code browser further:

displays the source code; and

tracks use of data elements for read and write accesses.

66. The software visualization environment of claim 65, wherein the code browser further:

highlights a variable in the source code; and

navigates through the source code.

67. The software visualization environment of 65, wherein the code browser further:

displays a list of all scalar and array variables, and

displays indexing patterns used in the code to access a selected array variable from the list.

68. The software visualization environment of 65, wherein the code browser further:

displays line numbers that cross reference to other visualization components.

69. The software visualization environment of claim 64, wherein the block-level abstract syntax tree viewer further:

displays a control structure of the source code, wherein the control structure comprises control blocks of the source code.

70. The software visualization environment of claim 69, wherein the block-level abstract syntax tree viewer further:

annotates the control blocks with information.

71. The software visualization environment of claim 70, wherein the information further comprises:

beginning and ending line numbers for each block; and

a list of variables read and written in each block.

72. The software visualization environment of claim 69, wherein the block-level abstract syntax tree viewer further:

expands and contracts the control blocks to display different levels of detail.

73. The software visualization environment of claim 64, wherein the program slice browser further:

displays a program slice as a directed graph comprising a plurality of nodes.

74. The software visualization environment of claim 73, wherein the nodes represent source code statements within a selected subroutine

75. The software visualization environment of claim 73, wherein the nodes represent variable references outside of a selected subroutine.

76. The software visualization environment of claim 73, wherein the nodes represent calls made to subroutines.

77. The software visualization environment of claim 73, wherein the program slice browser further:

displays the directed graph in a multi-column layout, wherein the nodes are positioned according to an order of corresponding statements in the source code.

78. The software visualization environment of claim 73, wherein the program slice browser further:

displays the directed graph in upside-down-tree layout, wherein the nodes are positioned according to a data-flow pattern.

79. The software visualization environment of claim 78, wherein the program slice browser further:

displays the layout based on user-selected parameters of width, height, and spread factor.

80. The software visualization environment of claim 78, wherein the program slice browser

further:

highlights edges emanating from a user-selected node.

81. The software visualization environment of claim 73, wherein the program slice browser further:

performs a block-level abstraction by representing statements belonging to a block by a single node in the directed graph.

82. The software visualization environment of claim 73, wherein the program slice browser further:

performs a procedure-level abstraction by displaying information regarding procedure calls.

83. The software visualization environment of claim 73, wherein the program slice browser further:

performs a graph-level abstraction by substituting a single node for a group of nodes in the directed graph, wherein the group of nodes is determined by a graph property.

84. The software visualization environment of claim 73, wherein the program slice browser further:

performs domain-specific semantic entity analysis.

85. The software visualization environment of claim 81, wherein the program slice browser further:

labels the single node with a block type and source code line numbers defining a boundary of the block.

86. The software visualization environment of claim 81, wherein the block comprises one of a group consisting of: an ordinary block, a call block, or a control block.

87. The software visualization environment of claim 81, wherein the single node retains a dependence relationship with the rest of the plurality of nodes in the directed graph.

88. The software visualization environment of claim 84, wherein the program slice browser further:

groups a plurality of nodes of the directed graph based on the domain-specific semantic entity analysis.

89. The software visualization environment of claim 88, wherein the semantic entity is a data element.

90. The software visualization environment of claim 88, wherein the semantic entity is a logical event.

91. The software visualization environment of claim 61, wherein the template viewer further:

displays a binding between a logical view of the source code and the source code.

92. The software visualization environment of claim 91, wherein the binding comprises a template, comprising:

abstract data structures; and

logical steps that manipulate the abstract data structures.

93. The software visualization environment of claim 64, further comprising:
a controller that cross-references information between the code browser, the block-level abstract syntax tree viewer, the program slice browser, and the template viewer.

94. The software visualization environment of claim 64, wherein the block-level abstract syntax tree viewer further:

adds instrumentation code to the source code to collect timing information.